Network Working Group
Internet-Draft
Updates: 4492 (if approved)
Intended status: Informational
Expires: July 12, 2014

S. Josefsson
SJD AB

January 8, 2014

Elliptic Curve Diffie-Hellman Key Agreement using Curve25519 for Transport Layer Security (TLS) draft-josefsson-tls-curve25519-02

Abstract

This document specifies the use of Curve25519 for key exchange in the Transport Layer Security (TLS) protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 12, 2014.

1. Introduction

In [Curve25519], a new elliptic curve function for use in cryptographic applications was specified. Curve25519 is a Diffie-Hellman function designed with performance and security in mind.

[RFC4492] defines the usage of elliptic curves for authentication and key agreement in TLS 1.0 and TLS 1.1, and these mechanisms are also applicable to TLS 1.2 [RFC5246]. The use of ECC curves for key exchange requires the definition and assignment of additional NamedCurve IDs. This document specify that value for Curve25519, as well as the minor changes in key selection and representation that are required to accomodate for Curve25519's slightly different nature.

This document only describes usage of Curve25519 for ephemeral key exchange (ECDHE), not for use with long-term keys embedded in PKIX certificates (ECDH_ECDSA and ECDH_ECDSA).

Curve25519 is not directly suitable for authentication, and thus not applicable for signing of e.g. PKIX certificates. See draft-josefsson-eddsa-ed25519 for a parallel effort.

2. Data Structures and Computations

[RFC4492] and related standards [SEC1] define an elliptic curve over GF(p) as the set of solutions of the equation y^2 = x^3 + a x + b (commonly referred to as a short Weierstrass equation) with both x and y in GF(p), plus the point at infinity. For each curve, a point G is fixed, generating a subgroup of large (prime) order N.

The Diffie-Hellman key exchange is then defined as follows: each party generates a random number 1 <= d < N (the private key), computes Q = d G (the public key). The parties exchange their public keys and compute the shared secret as Z = d Q_peer.

[RFC4492] defines formats for representing public keys during the exchange, and extensions for negotiating the format used by each party and the underlying curve used by both parties.

While retaining the same overall structure for the Diffie-Hellman key exchange, Curve25519 makes some changes to the way the curve equation is presented, private keys are selected and public keys exchanged, in order to ease secure and efficient implementations.

The following subsections describe the differences between using Curve25519 and the curves defined by RFC 4492 for key exchange in TLS.

2.1. Group negotiation and new NamedCurve type

Curve negotiation is the same for Curve25519 as for other curves, but is restricted to using the named_curve type in the ServerKeyEchange message: the explicit_prime type is only suited to curves in short Weierstrass form. This document adds a new NamedCurve value for Curve25519 as follows.

```
enum {
     curve25519(TBD1),
} NamedCurve;
```

The curve is suitable for use with DTLS [RFC6347].

Since Curve25519 is not designed to be used in signatures, clients who offer ECDHE_ECDSA ciphersuites and advertise support Curve25519 in the elliptic_curves ClientHello extension SHOULD also advertise support for at least one other curve, suitable for ECDSA. Servers MUST NOT select an ECDHE_ECDSA ciphersuite if the only common curve is Curve25519.

2.2. Private key generation

Private keys MUST be selected as specified in [Curve25519]. That is, private keys are 255-bits numbers with the following properties: the most significant bit (bit 254) is set, and the three least significants bits are cleared; the remaining bits (3 to 253 included) are chosen uniformly at random.

2.3. Public key representation

For ECDHE, public keys need to be transmitted in the ServerKeyExchange and ClientKeyExchange messages, both of which encode it as follows.

```
struct {
    opaque point <1..2^8-1>;
} ECPoint;
```

For short Weierstrass curves, the contents of ECPoint.point are defined by X9.62. For Curve25519, the ECpoint structure is the same, but the contents of ECPoint.point are encoded and interpreted as follows: each peer's public key is a number between 0 and 2^255 - 20 included, and ECPoint.point contains the 32 bytes string representing this number in big endian convention. (The receiving party can accept any 32 bytes string, interpreted as a 256 bits number, as public key: by design, no validation is needed.)

Note that ECPoint.point differs from the definition of public keys in [Curve25519] in two ways: (1) the byte-ordering is big-endian, wich is more uniform with how big integers are represented in TLS, and (2) there is an additional length byte (so ECpoint.point is actually 33 bytes), again for uniformity (and extensibility).

Since only one point format can be used with Curve25519, which is distinct from the formats used by short Weierstrass curves, the contents of the "Supported Point Formats" extension is irrelevant for this curve. Peers do not need to advertise support for the above point format in any way (nor check that the orther party supports it) when planning to use Curve25519 for key agreement: support for Curve25519 implies support for the above point format, which is tied to it.

2.4. Shared secret computation

As in the standard Elliptic Curve Diffie-Hellman protocol [SEC1], each party computes the shared secret by multiplying the peer's public value (seen as a point on the curve) by its own private value, except that in the case of Curve25519, only the x coordinate is computed. This is merely an internal detail since [RFC4492] specifies that only the x coordinate is used as the premaster secret anyway.

Again, in line with [RFC4492] and as a departure from the convention chosen in [Curve25519], the x coordinate is converted to a bytes string using big endian order. As in [RFC4492], leading zeros are preserved, so the premaster secret is always a 32 bytes string with Curve25519.

3. IANA Considerations

IANA is requested to assign numbers for Curve25519 listed in Section 2.1 to the Transport Layer Security (TLS) Parameters registry EC Named Curve [IANA-TLS] as follows.

Value	Description	DTLS-OK	Reference
TBD1	curve25519	Y	This doc

Table 1

4. Security Considerations

The security considerations of [RFC5246] and most of the security considerations of [RFC4492] apply accordingly.

Curve25519 was specifically designed so that secure, fast implementations are easier to produce. In particular, no validation of public keys is required, and point multiplication (using only the x coordinate) can be efficiently [EFD] computed with a Montgomery ladder using a constant number of operations (since the actual bit length of the private key is fixed), which avoids a number of side-channel attacks. However, in the fight against side-channel leaks, implementors should also pay attention to the following points:

1. In the Montgomery ladder, avoid branches depending on secret data (the individual bits of the secret key);

2. In the same place, avoid memory access patterns dependant on secret data;

3. Either avoid data-dependant branches and memory access patterns in the underlying field arithmetic (that is, the bignum arithmetic, including the mod 2^255-19 operation) or randomize projective (that is, x/z) coordinates by multiplying both x and z with the same 256-bit value, chosen at random.

Some of the curves defined in [RFC4492], namely all whose name ends with r1 or r2, have been advertised as pseudo-randomly chosen, but the lack of verifiability of the seeds has raised concerns that the those curves might be weaker than expected aginst some attackers. The Koblitz curves (those whose name end with k1) of [RFC4492] do not suffer from this problem, but are char2 curves and there seems to be a consensus that curves over prime fields are a safer bet against future progress in discrete log computation. The Brainpool curves [RFC7027] are prime curves generated in a fully verifiable pseudo-random way to avoid manipulation concerns, but do not perform as well due to the use of pseudo-random primes. Curve22519 is also chosen in a fully verifiable way, but offers better performances (better than the curves form [RFC4492]) while facilitating secure implementations as mentioned above. See also [SafeCurves].

5. References

5.1. Normative References

[Curve25519]
Bernstein, J., "Curve25519: new Diffie-Hellman speed records", WWW http://cr.yp.to/ecdh/curve25519-20060209.pdf, February 2006.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

5.2. Informative References

[IANA-TLS] Internet Assigned Numbers Authority, "Transport Layer Security (TLS) Parameters", <http://www.iana.org/assignments/tls-parameters/tls-parameters.xml>.

[SafeCurves] Bernstein, D. and T. Lange, "SafeCurves: choosing safe curves for elliptic-curve cryptography.", January 2014, <http://safecurves.cr.yp.to/>.

[EFD] Bernstein, D. and T. Lange, "Explicit-Formulas Database: XZ coordinates for Montgomery curves", January 2014, <http://www.hyperelliptic.org/EFD/g1p/auto-montgom-xz.html>.

[RFC7027] Merkle, J. and M. Lochter, "Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)", RFC 7027, October 2013.

[SEC1] Certicom Research, , "Standards for Efficient Cryptography (SEC) 1", September 2000.

Appendix A. Test vectors

This section provides some test vectors for example Diffie-Hellman key exchanges using Curve25519. The following notations are used:

d_A the secret key of party A

x_A the public key of party A

d_B the secret key of party B

x_B the public key of party B

x_S the shared secret that results from completion of the Diffie-Hellman computation, i.e., the hex representation of the pre-master secret.

The field elements x_A, x_B, and x_S are represented as hexadecimal values using the FieldElement-to-OctetString conversion method specified in [SEC1].

```
d_A = 5AC99F33632E5A768DE7E81BF854C27C46E3FBF2ABBACD29EC4AFF51
      7369C660
d_B = 47DC3D214174820E1154B49BC6CDB2ABD45EE95817055D255AA35831
      B70D3260
x_A = 057E23EA9F1CBE8A27168F6E696A791DE61DD3AF7ACD4EEACC6E7BA5
      14FDA863
x_B = 6EB89DA91989AE37C7EAC7618D9E5C4951DBA1D73C285AE1CD26A855
      020EEF04
x_S = 61450CD98E36016B58776A897A9F0AEF738B99F09468B8D6B8511184
      D53494AB
```

Author's Address

Simon Josefsson
SJD AB

Email: simon@josefsson.org